VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Abhinav Sanjay (1BM23CS009)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Object Oriented Java Programming (23CS3PCOOJ)" carried out by **Abhinav Sanjay (1BM23CS009),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

Lab faculty Incharge :Geetha N Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:
https://github.com/AbhinavSanjay24/Java

Program 1

Implement Quadratic Equation

Algorithm:

- **Step 1:** Read the coefficients a, b, and c from the user input.
- **Step 2:** Calculate the discriminant D using the formula D=b^2−4ac
- **Step 3:** Check the value of D:
    - If D=0, the roots are real and equal, and you compute them using –b/2a.
    - If D>0, the roots are real and distinct, and you compute them using the formulas for r1 and r2
    - If D<0, the roots are imaginary, and you compute the real and imaginary parts, then display the complex roots in the form of r1+r2i and r1−r2i.
- **Step 4:** Output the type of roots and the root values to the user.

9/10/24

LAB - 1

a) Develop a java program that prints all real solutions to the quadratic equation $ax^2 + bx + c = 0$. Read a, b, c and use the quadratic formula. If the discriminate $b^2 - 4ac$ is negative, display a message stating that there are no real solutions.

```
import java.util.*;
class Quad {
Scanner sc = new Scanner (System.in);
int a, b, c, d;
double r1, r2, d_sq;

void input () {
System.out.println ("Enter coefficients a, b, c:");
a = sc.nextInt ();
b = sc.nextInt ();
c = sc.nextInt ();
}

Void calc () {
int d = b*b - 4*a*c;
System.out.println (d);
if (d==0) {
r1 = -b/(2.0*a);
System.out.println ("Roots are real and equal");
System.out.println ("Root 1 = "+r1+"\n Root 2 = "+r1);
}
```

```
else if (d>0) {
d_sq = Math.sqrt(d);
r1 = (-b + d_sq)/(2.0*a);
r2 = (-b + d_sq)/(2.0*a);          real and distinct
System.out.println("Roots are imaginary");
System.out.println("Root1 = "+r1+"\n Root2 = "+r2);
}
else {
d_sq = Math.sqrt(-d);
r1 = -b/(2.0*a);
r2 = d_sq/(2.0*a);
System.out.println("Roots are imaginary");
System.out.println("Root1 = "+r1+" + "+r2+"i" +
    "\n Root 2 = "+r1+" - "+r2+"i");
}}}

class Quadratic {
public static void main (String[] args) {
Quad quad = new Quad();
quad.input();
quad.calc();
}
}
```

Output

enter coefficients a, b, c

1

5

6

Roots are real and distinct

Root 1 = 2.0

Root 2 = 2.0

enter coefficients a, b, c

1

1

1

Roots are imaginary

Root 1 = -0.5 + 0.8660254037844386 i

Root 2 = -0.5 + 0.8660254037844386 i

Enter coefficient a, b, c

4

12

9

Roots are real and equal

Root 1 = -1.5

Root 2 = -1.5

Output seen

9/10/24

Code:

```
import java.util.*;

class Quad {
    Scanner sc = new Scanner(System.in);
    int a, b, c, d;
    double r1, r2, d_sq;

    // Method to input coefficients
    void input() {
        System.out.println("Enter coefficients a, b, c:");
        a = sc.nextInt();
        b = sc.nextInt();
        c = sc.nextInt();
    }

    // Method to calculate the roots based on discriminant
    void calc() {
        int d = b * b - 4 * a * c;
        System.out.println(d);

        if (d == 0) {
            r1 = -b / (2.0 * a);
            System.out.println("Roots are real and equal");
            System.out.println("Root 1 = " + r1 + "\nRoot 2 = " + r1);
        } else if (d > 0) {
            d_sq = Math.sqrt(d);
            r1 = (-b + d_sq) / (2.0 * a);
            r2 = (-b - d_sq) / (2.0 * a);
            System.out.println("Roots are real and distinct");
            System.out.println("Root 1 = " + r1 + "\nRoot 2 = " + r2);
        } else {
            d_sq = Math.sqrt(-d);
            r1 = -b / (2.0 * a);
            r2 = d_sq / (2.0 * a);
            System.out.println("Roots are imaginary");
            System.out.println("Root 1 = " + r1 + " + " + r2 + "i" + "\nRoot 2 = " + r1 + " -
" + r2 + "i");
        }
    }
```

```
}

class Quadratic {
   public static void main(String[] args) {
      System.out.println("Name: Abhinav Sanjay \nUSN: 1BM23CS009");
      Quad quad = new Quad();
      quad.input();
      quad.calc();
   }
}
```

Output:

```
D:\Abhinav3A\Java>java Quadratic
Name: Abhinav Sanjay
USN: 1BM23CS009
Enter coefficients a,b,c:
1
5
6
1
Roots are real and distinct
Root 1 = -2.0
Root 2 = -3.0

D:\Abhinav3A\Java>java Quadratic
Name: Abhinav Sanjay
USN: 1BM23CS009
Enter coefficients a,b,c:
1
1
1
-3
Roots are imaginary
Root 1 = -0.5 + 0.8660254037844386i
Root 2 = -0.5 - 0.8660254037844386i

D:\Abhinav3A\Java>java Quadratic
Name: Abhinav Sanjay
USN: 1BM23CS009
Enter coefficients a,b,c:
4
12
9
0
Roots are real and equal
Root 1 = -1.5
Root 2 = -1.5

D:\Abhinav3A\Java>
```

Program 2

Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

Algorithm:

1. Start the program.
2. Input the number of students.
3. For each student:
    - Input the name and USN.
    - For Semester 1:
        - Input the credits and marks for 8 subjects.
        - Calculate SGPA:
            - For each subject:
                - If the mark is not 100, compute the grade by (mark + 10) / 10.
                - If the mark is 100, assign the grade as 10.
                - Calculate the total weighted sum: sum = sum + (grade * credits).
                - Calculate the total credits: total_credits = total_credits + credits.
            - Compute SGPA by dividing the sum of weighted grades by the total credits.
            - Display SGPA for Semester 1.
    - For Semester 2:
        - Input the credits and marks for 8 subjects.
        - Calculate SGPA for Semester 2 following the same steps.
        - Display SGPA for Semester 2.
    - Calculate CGPA as the average of the SGPA for both semesters: \
    - Display CGPA for the 1st year.
4. End the program.

16/10/24

LAB - 2

2) Develop a java program to create a class Student with members USN, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

```
import java.util.Scanner;
class Student
{
  String name;
  String USN;
  int credits[] = new int[5];
  int marks[] = new int[5];
  double sgpa = 0.0;
  double cgpa;
  int grade[] = new int[5];

double calculate(int m[], int c[])
{
  int j;
  double sum = 0.0;
  int div = 0;
  for(j = 0; j < 5; j++)
  {
    if(m[j] != 100)
    {
      grade[j] = (m[j] + 10)/10;
    }
    else
```

```
{
    grade [j] = 10;
}
div = credits [j] + div;
sum = sum + (grade [j]) * credits [j]);
System. out. print ln ("Grade for subject"
+ (j + 1) + ":" + grade [j]);
}
sgpa = sum / div;
System. out. print ln ("SGPA:" + sgpa);
return sgpa;
}

double calcgpa (double sgpa1, double sgpa2)
{
    cgpa = (sgpa1 + sgpa2) / 2;
    return cgpa;
}

void input ()
{
Scanner sc = new Scanner (System. in);
System. out. print ln ("Enter subject credit for semester:");
int i;
for (i = 0; i < 5; i++)
{
  credits [i] = sc. next Int ();
}
```

```
System.out.println("Enter marks for suliject :");
for (i=0; i<5; i++)
 {
  marks [i] = sc.nextInt();
 }
}

Public static void main (String args[]) {
Scanner sc1 = new Scanner (System.in);
System.out.println("Enter number of students");
int n = sc1.nextInt();
Student obj[] = new Student [n];
int K;
for( K=0; K<n; K++)
{
obj[K] = new Student();
System.out.println("Enter student name.");
name = sc1.~~nn~~;
System.out.println("Enter student USN");
USN = sc1.nextline();
obj[K].input();
System.out.println("Semester 1");
double result = obj[K].calculate (obj[K].marks,
obj[K].credits);
System.out.println("1st Semester SGPA for" + obj[K].name
+ "(" + obj[K].USN + ") is :" + result);
System.out.println("Semester 2");
obj[K].input();
```

```
double result2 = obj[K].calculate(obj[K].marks,
obj[K].credits);
System.out.println("2nd Semester SGPA for"
+ obj[K].name + "(" + obj[K].USN + ") is :" + result2);
System.out.println("CGPA for 1st year is :"
+ obj[K].calcgpa(result, result2));
}
}
}
```

Seen
16/10/24

Output

```
Enter number of students
1
Enter student name
Abhinav
Enter student USN
1
Enter subjects credits for semester
3
3
3
3
3
3
3
3
3
```

Enter marks
85
90
91
90
89
78
70
95
Semester 1
Grade for subject 1 : 9
Grade for subject 2 : 10
Grade for subject 3 : 10
Grade for subject 4 : 10
Grade for subject 5 : 9
Grade for subject 6 : 8
Grade for subject 7 : 8
Grade for subject 8 : 10
SGPA = 9.25
1st Semester SGPA for null (null) is : 9.25
Semester 2
Enter credits
3
3
3
3
3
3
3
3

Enter marks

90

98

95

96

87

85

80

89

Grade for subject 1 : 10

Grade for subject 2 : 10

subject 3 : 10

subject 4 : 10

subject 5 : 9

subject 6 : 9

subject 7 : 9

subject 8 : 9

SGPA : 9.5

2nd ~~semester~~ SGPA for null (null) is : 9.5

CGPA for first year is : 9.375

O/P seen

GL

16/10/24

Code:

```
import java.util.Scanner;

class Student {
    // Member variables
    String name;
    String usn;
    int credits[] = new int[8];
    int marks[] = new int[8];
    double sgpa = 0.0;
    double cgpa;
    int grade[] = new int[8];

    // Method to calculate SGPA
    double calculate(int m[], int c[]) {
        double sum = 0.0;
        int div = 0;

        for (int j = 0; j < 8; j++) {
            // Calculate grade for each subject
            if (m[j] != 100) {
                grade[j] = (m[j] + 10) / 10;
            } else {
                grade[j] = 10;
            }
            div = credits[j] + div;
            sum = sum + (grade[j] * credits[j]);

            // Debugging output for grade calculation
            System.out.println("Grade for subject " + (j + 1) + ": " + grade[j]);
        }

        // Calculate SGPA
        sgpa = sum / div;
        System.out.println("SGPA: " + sgpa);
        return sgpa;
    }

    // Method to calculate CGPA
    double calcgpa(double sgpa1, double sgpa2) {
        cgpa = (sgpa1 + sgpa2) / 2;
```

```
    return cgpa;
}

// Method to input credits and marks for the semester
void input() {
    Scanner sc = new Scanner(System.in);

    // Input credits
    System.out.println("Now enter subject credits for semester:");
    for (int i = 0; i < 8; i++) {
        credits[i] = sc.nextInt();
    }

    // Input marks
    System.out.println("Now enter subject marks for semester:");
    for (int i = 0; i < 8; i++) {
        marks[i] = sc.nextInt();
    }
}

public static void main(String args[]) {
    Scanner sc1 = new Scanner(System.in);

    // Input number of students
    System.out.println("Enter number of students: ");
    int n = sc1.nextInt();

    // Create array of Student objects
    Student obj[] = new Student[n];

    // Loop through each student
    for (int k = 0; k < n; k++) {
        obj[k] = new Student();

        // Input student details
        System.out.println("Enter Student name: ");
        obj[k].name = sc1.next();

        System.out.println("Enter Student USN: ");
        obj[k].usn = sc1.next();

        // Input semester details and calculate SGPA for Semester 1
```

```
            System.out.println("Semester 1");
            obj[k].input();
            double result = obj[k].calculate(obj[k].marks, obj[k].credits);
            System.out.println("1st Semester SGPA for " + obj[k].name + " (" + obj[k].usn
+ ") is: " + result);

            // Input semester details and calculate SGPA for Semester 2
            System.out.println("Semester 2");
            obj[k].input();
            double result2 = obj[k].calculate(obj[k].marks, obj[k].credits);
            System.out.println("2nd Semester SGPA for " + obj[k].name + " (" +
obj[k].usn + ") is: " + result2);

            // Calculate and display CGPA for the year
            System.out.println("CGPA for 1st year is: " + obj[k].calcgpa(result, result2));
        }
    }
}
```

Output:

```
D:\Abhinav3A\Java>java Student
Enter number of students:
3
Enter Student name:
A
Enter Student USN:
1
Now enter subject credits for semester:
3
3
3
3
3
3
3
3
Now enter subject marks for semester:
85
90
91
90
89
78
70
95
Semester 1
Grade for subject 1: 9
Grade for subject 2: 10
Grade for subject 3: 10
Grade for subject 4: 10
Grade for subject 5: 9
Grade for subject 6: 8
Grade for subject 7: 8
Grade for subject 8: 10
SGPA: 9.25
1st Semester SGPA for null (null) is: 9.25
Semester 2
Now enter subject credits for semester:
5
5
5
5
5
5
5
5
Now enter subject marks for semester:
90
98
95
96
87
85
80
```

Program 3

Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString( ) method that could display the complete details of the book. Develop a Java program to create n book objects.

Algorithm:

1) Start the program.

2) Input the number of books (n) from the user:

- Prompt the user: "Enter the number of books: ".
- Read the input value into n.

3) Create an Array of Books:

- Declare an array Book[] books = new Book[n].

4) For each book from 1 to n:

- Input book details:
  - Prompt and input the name of the book.
  - Prompt and input the author of the book.
  - Prompt and input the price of the book.
  - Prompt and input the number of pages of the book.
- Create a new Book object using the entered details.
  - Store the object in books[i] for the corresponding book.

5) Display Book Details:

- For each book in the books array, print the details using the toString() method.

6) End the program.

23/10/24

LAB - 3

Q) Create a class Book which contains four members : name, author, price, numPages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include toString() method that could display the complete details of the book. Create n books.

```
import java.util.scanner

class Book {
    private string name;
    private string author;
    private double price;
    private int numPages;

public Book (String name, String author, double price,
int numPages) {

    this.name = name;
    this.author = author;
    this.price = price;
    this.numPages = numPages;
}

public void setName (String name) {
    this.name = name;
```

```
public void setAuthor(String author){
    this.author = author;
}

public void setPrice(double price){
    this.price = price;
}

public void setNumPages(int numPages){
    this.numPages = numPages;
}

public String getName(){
    return name;
}

public String getAuthor(){
    return Author;
}

public double getPrice(){
    return Price;
}

public int getnumPages(){
    return numPages;
}
```

```
public String toString() {
    return "Book Name:" + name + ", Author:" + author +
    ", Price:" + price + ", Number of Pages:" + numPages;
}
}

public class BookMain {
    public static void main (String[] args) {
        Scanner scanner = new Scanner (System.in);
        System.out.print ("Enter no. of books:");
        int n = scanner.nextInt();
        scanner.nextLine();

        Books[] books = new Book[n];

        for (int i = 0; i < n; i++) {
        System.out.println ("Enter book details " + (i+1) + ":");
        System.out.println ("Name");
        String name = scanner.nextLine();
        System.out.print ("Author");
        String author = scanner.nextLine();
        System.out.print ("Price");
        double price = scanner.nextDouble();
        System.out.print ("No. of Pages");
        int numPages = scanner.nextInt();
        scanner.nextLine();

        books[i] = new Book (name, author, price, numPages);
}
```

```
System.out.println("In Book Details ");
  for(Book book : books) {
    System.out.println(book);
}
 scanner.close();
}
}
```

Outputs

```
Enter number of books : 3
Enter details for book 1 :
Name : ABC
Author : Robert
Price : 500
Number of pages : 400

Enter details for book 2 :
Name : ~~ABC~~ DEF
Author : Chetan
Price : 400
Number of pages : 300

Enter details for book 3 :
Name : XYZ
Author : Ruskin
Price : 300
Number of pages : 200
```

Book Details :

Book name : ABC , Author : Robert, Price : 500.0,
Number of pages : 400

Book name : Chetan , Author : Chetan , Price : 400.0,
Number of pages : 600

Book name : ~~Ruskin~~ XYZ , Author : Ruskin , Price 300.0,
Number of pages : 200

Seen
Sk
03/10/24

Code:

```
import java.util.Scanner;

class Book {
    // Private member variables
    private String name;
    private String author;
    private double price;
    private int numPages;

    // Constructor to initialize book details
    public Book(String name, String author, double price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

    // Setter methods
    public void setName(String name) {
        this.name = name;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void setNumPages(int numPages) {
        this.numPages = numPages;
    }

    // Getter methods
    public String getName() {
        return name;
    }

    public String getAuthor() {
```

```
        return author;
    }

    public double getPrice() {
        return price;
    }

    public int getNumPages() {
        return numPages;
    }

    // Override toString method to display book details
    @Override
    public String toString() {
        return "Book Name: " + name + ", Author: " + author + ", Price: " + price + ",
Number of Pages: " + numPages;
    }
}

public class BookMain {
    public static void main(String[] args) {
        // Displaying student details
        System.out.println("Name: Abhinav Sanjay\nUSN : 1BM23CS009");

        // Creating scanner object for input
        Scanner scanner = new Scanner(System.in);

        // Input the number of books
        System.out.println("\nEnter the number of books: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character left by nextInt()

        // Array to hold book objects
        Book[] books = new Book[n];

        // Loop to input details for each book
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for book " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Author: ");
            String author = scanner.nextLine();
```

```
            System.out.print("Price: ");
            double price = scanner.nextDouble();
            System.out.print("Number of Pages: ");
            int numPages = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character left by nextInt()

            // Create a new Book object and store it in the array
            books[i] = new Book(name, author, price, numPages);
        }

        // Display book details
        System.out.println("\nBook Details:");
        for (Book book : books) {
            System.out.println(book);
        }

        // Close the scanner to prevent resource leaks
        scanner.close();
    }
}
```

Output:

```
D:\Abhinav3A\Java>java BookMain
Name: Abhinav Sanjay
USN : 1BM23CS009

Enter the number of books:
3
Enter details for book 1:
Name: ABC
Author: Robert Frost
Price: 500
Number of Pages: 300
Enter details for book 2:
Name: DEF
Author: Chetan Bhagat
Price: 350
Number of Pages: 200
Enter details for book 3:
Name: Ruskin Bond
Author: XYZ
Price: 400
Number of Pages:
500

Book Details:
Book Name: ABC, Author: Robert Frost, Price: 500.0, Number of Pages: 300
Book Name: DEF, Author: Chetan Bhagat, Price: 350.0, Number of Pages: 200
Book Name: Ruskin Bond, Author: XYZ, Price: 400.0, Number of Pages: 500
```

Program 4

Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea( ). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea( ) that prints the area of the given

Algorithm:

1) Start the Program.
2) Create an abstract class called Shape with:

   - Two variables: dimension1 and dimension2.
   - An abstract method printArea() to be implemented by subclasses.

3) Create the Rectangle class:

   - Initialize dimensions length and width.
   - Calculate and print the area of the rectangle.

4) Create the Triangle class:

   - Initialize dimensions base and height.
   - Calculate and print the area of the triangle.

5) Create the Circle class:

   - Initialize radius.
   - Calculate and print the area of the circle.

6) In ShapeTest class:

   - Create a Rectangle object and call printArea().
   - Create a Triangle object and call printArea().
   - Create a Circle object and call printArea().

7) End the Program

30/10/24

LAB - 4

1Q) Develop a java program to create an abstract class name Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

```
abstract class Shape {
  protected int dimension1;
  protected int dimension2;

  public abstract void Area();
}
class Rectangle extends Shape {
  public Rectangle(int length, int width) {
    this.dimension1 = length;
    this.dimension2 = width; }
public void ~~print~~ Area(){
  int area = dimension1 * dimension2;
  System.out.print("Area of rectangle" + area); }}

class triangle extends Shape {
  public triangle(int base, int height){
    this.dim1 = base;
    this.dim2 = height; }
```

```
public void Area() {
double area = 0.5 * dim1 * dim2;
System.out.println("Area of triangle" + area); }}

class Circle extends Shape {
  public Circle (int radius) {
    this.dim1 = radius; }

public void Area() {
  double Area = Math.Pi * dim1 * dim2;
  System.out.println("Area of circle" + area); }}

public class ShapeTest {
  public class void main (String [] args) {
  Shape rectangle = new rectangle (5,3);
  rectangle.Area();
  Shape triangle = new triangle (4,6);
  triangle.Area();
  Shape circle = new circle (7);
  circle.Area(); }}
```

Output:

Area of rectangle : 15
Area of triangle : 12.0
Area of circle : 153.938

Code:

```
abstract class Shape {
    protected int dimension1; // Could represent length, base, or radius
    protected int dimension2; // Could represent width or height, or could be unused for
Circle

    // Abstract method to print area, to be implemented by subclasses
    public abstract void printArea();
}

class Rectangle extends Shape {
    // Constructor to initialize the dimensions of the rectangle
    public Rectangle(int length, int width) {
        this.dimension1 = length;
        this.dimension2 = width;
    }

    // Override the printArea method to calculate and print the area of the rectangle
    public void printArea() {
        int area = dimension1 * dimension2;
        System.out.println("Area of Rectangle: " + area);
    }
}

class Triangle extends Shape {
    // Constructor to initialize the dimensions of the triangle
    public Triangle(int base, int height) {
        this.dimension1 = base;
        this.dimension2 = height;
    }

    // Override the printArea method to calculate and print the area of the triangle
    public void printArea() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape {
    // Constructor to initialize the radius of the circle
```

```
    public Circle(int radius) {
        this.dimension1 = radius; // Only one dimension is needed for Circle
    }

    // Override the printArea method to calculate and print the area of the circle
    public void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}

public class ShapeTest {
    public static void main(String[] args) {
        // Create and test Rectangle object
        Shape rectangle = new Rectangle(5, 3);
        rectangle.printArea();

        // Create and test Triangle object
        Shape triangle = new Triangle(4, 6);
        triangle.printArea();

        // Create and test Circle object
        Shape circle = new Circle(7);
        circle.printArea();
    }
}
```

Output:

```
D:\Abhinav3A\Java>java ShapeTest
Name: Abhinav Sanjay
USN:1BM23CS009
Area of Rectangle: 15
Area of Triangle: 12.0
Area of Circle: 153.93804002589985
```

Program 5

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

a)Accept deposit from customer and update the balance.
b) Display the balance.
c) Compute and deposit interest
d) Permit withdrawal and update the balance
e) Check for the minimum balance, impose penalty if necessary and update the balance.

Algorithm:

1) Start the Program.
2) Prompt User to Choose Account Type:

   - Ask the user to select whether they want to open a Savings or a Current account.
   - Create an account object based on the user's choice.

3) Show Menu for Operations:

   - Display the options: Deposit, Withdraw, Show Balance, Compute Interest, Exit.

4) Perform Operations Based on User Input:

   - Deposit: Ask the user for a deposit amount and add it to the balance.
   - Withdraw: Ask the user for a withdrawal amount and subtract it from the balance. If the balance is below the withdrawal limit in a CurrentAccount, deduct a service charge.
   - Show Balance: Display the current account balance.
   - Compute Interest: For SavingsAccount, calculate and display the interest and add it to the balance.

5) End the Program.

30/10/24

LAB 5

Q) Develop a java program to create a class Bank that maintains two kinds of account - savings and current. Savings account provides compound interest and withdrawal facilities but no cheque book facility. Current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

```
import java.util.Scanner;
import java.lang.Math;
class Bank {
    int accountNo;
    double balance;
    Bank (int accountNo) {
        this.accountno = accountno;
        this.balance = 0;
    }
  Void deposit (double depAmount) {
    this.balance += depAmount;
  }
  Void withdraw (double withAmount) {
    this.balance -= withamount;
  }
  double interest (double rate, int time) {
   System.out.println ("Interest is not applicable
   in current amount account");
   return 0.0; }}
```

```
class SavingsAccount extends Bank {
    SavingsAccount (int accountNo) {
        super (accountno);
}
double interest (double rate, int time) {
    double interest = (balance * Math.pow ((1 +
        (rate/100)), time)) - balance;
        balance += interest;
        return interest; }
}
class CurrentAccount extends Bank {
    static double withdrawLimit = 1000;
    CurrentAccount (int accountNo) {
        super (accountno);
}
    public void withdraw (double withAmount) {
        super.balance -= withAmount;
        if (balance < withdrawLimit) {
            System.out.println ("Withdraw limit reached
            - Deducting service charge");
            balance -= 100; }
    }
}
class Bank {
    public static void main (String [] args) {
    System.out.println
    double amount;
    Scanner sc = new Scanner (System.in);
    System.out.println ("1. Open savings acc \n 2.
    Open current acc \n Enter choice");
```

```
int choice = SC.nextInt();
Bank acc;
if (choice == 1) {
        acc = new SavingsAccount (101); }
else {
        acc = new CurrentAccount (201); }
System.out.println(1. Deposit \n 2. Withdraw 3. Show
  balance \n 4. Compute interest \n 5. Exit);
while (true) {
  System.out.print("Enter choice");
  choice = SC.nextInt();
  switch (choice){
    case 1: System.out.print("Enter deposit amt")
            amount = sc.nextDouble();
            acc.Deposit(amount);
            break;
    case 2: System.out.print("Withdraw amount")
            amount = SC.nextDouble();
            acc.withdraw(amount);
            break;
    case 3: System.out.println("Balance: + acc.balance);
            break;
    case 4: System.out.println("Interest: acc.interest(5,1));
            break;
    default: System.exit(0); }
  }
 }
}
```

Output

1) Open Savings Account
2) Open Current Account

Enter choice : 1
~~Enter deposit amount : 5000~~
~~Enter choice :~~

1) Deposit
2) Withdrawal
3) Show balance
4) Compute interest
5) Exit

Enter choice : 1
Enter deposit amount : 5000
Enter choice : 2
Enter withdrawal amount : 1000
Enter choice : 3
Balance is 4000.0
Enter choice : 4
Interest is 200.0

Seen
20/11/24

Code:

```
import java.util.Scanner;
import java.lang.Math;

class Bank {
    int accountNo;
    double balance;

    Bank(int accountNo) {
        this.accountNo = accountNo;
        this.balance = 0;
    }

    void deposit(double depAmount) {
        this.balance += depAmount;
    }

    void withdraw(double withAmount) {
        this.balance -= withAmount;
    }

    double interest(double rate, int time) {
        System.out.println("Interest is not applicable in current account");
        return 0.0;
    }
}

class SavingsAccount extends Bank {
    SavingsAccount(int accountNo) {
        super(accountNo);
    }

    double interest(double rate, int time) {
        double interest = (balance * Math.pow((1 + (rate / 100)), time)) - balance;
        balance += interest;
        return interest;
    }
}
```

```
class CurrentAccount extends Bank {
   static double withdrawLimit = 1000;

   CurrentAccount(int accountNo) {
      super(accountNo);
   }

   public void withdraw(double withAmount) {
      super.balance -= withAmount;
      if (balance < withdrawLimit) {
         System.out.println("Withdraw Limit Reached - Deducting Service Charge");
         balance -= 100;
      }
   }
}

class Run {
   public static void main(String[] args) {
      System.out.println("Abhinav Sanjay 1BM23CS009");
      double amount;
      Scanner sc = new Scanner(System.in);

      System.out.print("1. Open Savings Account\n2. Open Current Account\n\nEnter
Choice: ");
      int choice = sc.nextInt();
      Bank acc;

      if (choice == 1) {
         acc = new SavingsAccount(101);
      } else {
         acc = new CurrentAccount(201);
      }

      System.out.println("1. Deposit\n2. Withdraw\n3. Show Balance\n4. Compute
Interest\n5. Exit\n");
      while (true) {
         System.out.print("Enter Choice: ");
         choice = sc.nextInt();
         switch (choice) {
            case 1:
               System.out.print("Enter deposit amount: ");
               amount = sc.nextDouble();
```

```
                acc.deposit(amount);
                break;
            case 2:
                System.out.print("Enter withdraw amount: ");
                amount = sc.nextDouble();
                acc.withdraw(amount);
                break;
            case 3:
                System.out.println("The balance is " + acc.balance);
                break;
            case 4:
                System.out.println("The interest is " + acc.interest(5, 1));
                break;
            default:
                System.exit(0);
        }
    }
  }
}
```

Output:

```
Abhinav Sanjay 1BM23CS009
1. Open Savings Account
2. Open Current Account

Enter Choice:1
1. Deposit
2. Withdraw
3. Show Balance
4. Compute Interest
5. Exit

Enter Choice: 1
Enter deposit amount: 5000
Enter Choice: 2
Enter withdraw amount: 1000
Enter Choice: 3
The balance is 4000.0
Enter Choice: 4
The interest is 200.0
Enter Choice: 5

C:\Abhinav 3A>java Run
Abhinav Sanjay 1BM23CS009
1. Open Savings Account
2. Open Current Account

Enter Choice:2
1. Deposit
2. Withdraw
3. Show Balance
4. Compute Interest
5. Exit

Enter Choice: 1
Enter deposit amount: 6000
Enter Choice: 2
Enter withdraw amount: 1000
Enter Choice: 3
The balance is 5000.0
Enter Choice: 4
Interest is not applicable in current account
The interest is 0.0
```

Program 6

Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses. CIE package should contain internal.class and student.class SEE package should contain External.class

Algorithm:

1) **Start the Program**:

- Display the name and USN of the student.

2) **Initialize Scanner for Input**:

- Create a Scanner object to read input from the user.

3) **Input Number of Students**:

- Prompt the user to enter the number of students (n).
- Create two arrays of size n:
  - One for storing Internal marks of students.
  - One for storing External marks of students.

4) **Input Details for Each Student**:

- Loop through each student (from 1 to n):
  - Prompt the user to enter the name of the student.
  - Prompt the user to enter the internal marks for 5 courses.
  - Store the entered internal marks in an array internalMarks[].
  - Prompt the user to enter the external marks for 5 courses.
  - Store the entered external marks in an array externalMarks[].
  - Create an Internal object and store it in the internalStudents[] array.
  - Create an External object and store it in the externalStudents[] array.

5) **Display Final Marks for All Students**:

- Loop through each student (from 1 to n):
  - Retrieve and print the student's name.

- Retrieve and print the internal marks for the student.
- Retrieve and print the external marks for the student.
- Calculate and print the final marks for each course:
  - For each course (5 courses), calculate the sum of internal and external marks.
  - Print the final marks for each course.

6) **Close the Scanner**:

- Close the Scanner object to free up resources.

7) **End the Program**.

13/11/24

LAB - 6

Q) Create a package CIE which has 2 classes - Student and Internals. The class Student has members like USN, name, sem. The class internals derived from Student has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

// Student.java

```
package CIE;
public class Student {
    protected String name;
    protected int [] marks;
    public Student (String name) {
    this.name = name;
    this.marks = new int [5];
}
public String getName () {
    return name;}
public void setMarks ( int [] marks) {
    this.marks = marks;
}
```

```
    public int [] getMarks () {
        return marks ; }
}

// Internal.java

package CIE ;
public class Internal extends Student {
    int [] internalMarks ;
    public Internal ( String name , int [] internalMarks){
        super (name) ;
        this . internalMarks = internalMarks ;
        this . setMarks ( internalMarks); }
}

// External.java
package SEE ;
import CIE.Student ;
public class External extends Student {
    int [] externalMarks ;
    public External ( String name, int [] externalMarks ){
        super (name);
        this . externalMarks = externalMarks;
        this . setMarks ( externalMarks ); }
}

import CIE.internal;
import SEE.external;
import java.util.Scanner;
```

```
public class Main {
    public static void main (String [] args) {
    System.out.println
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter no. of students");
    int n = sc.nextInt ();
    sc.nextline ();
    Internal [] internal Students = new Internal [n];
    External [] external Students = new External [n];
    for ( int i = 0 ; i < n ; i ++) {
    System.out.print ("Enter name" + (i+1) + " :");
    String name = sc.nextline ();
    System.out.print ("Enter marks");
    int [] internal Marks = new int [5];
    for ( int j = 0; j < 5 ; j++) {
      internal marks [j] = sc.nextline ();
}
sc.nextLine ();
System.out.print ("Enter external marks")
 int [] external marks = new int [5];
 for ( int j = 0 ; j < 5 ; j ++) {
   System.out.print ("Enter marks");
   external marks [j] = sc.nextInt ();
}
sc.nextLine ();
internal Students[i] = new Internal (name, internal Marks);
external Students [i] = new External (name, external marks);
}
```

```
System. out. print ("Final marks");
for (int i = 0; i < n ; i++) {
  int [] internal marks = interal Student [i]. get marks ();
  int [] external mark = external Student [i]. get mark ();
System. out. print ("Internal marks")
  for ( int mark : internal marks ) {
    System. out. println( mark + " ");
}
System. out. println ("External marks");
  for ( int mark : external marks ) {
    System. out. println ("E mark + " ");
}
System. out. println ("Final marks");
for ( int j = 0; j < 5; j++) {
  int final mark = internal mark [j] + external marks [j]
System. out. print ( final mark + " ");
}
sc. close (); }
}
```

Output :

Enter number of students : 1

Enter name of student 1 : A

Enter internal marks (5 courses) for A :

40

48

46

25

44

Enter external marks (5 courses) for A :

42

45

47

46

48

Final marks of all students :

Student : A

Internal marks : 40 48 46 25 44

External marks : 42 45 47 46 48

Final marks : 82 93 93 71 92

Code:

```
import CIE.Internal;
import SEE.External;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        System.out.println("Abhinav Sanjay 1BM23CS009");
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of students: ");
        int n = sc.nextInt();
        sc.nextLine();
        Internal[] internalStudents = new Internal[n];
        External[] externalStudents = new External[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter the Name of student " + (i + 1) + ": ");
            String name = sc.nextLine();
            System.out.println("Enter Internal Marks (5 courses) for " + name + ": ");
            int[] internalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                internalMarks[j] = sc.nextInt();
            }
            sc.nextLine();
            System.out.println("Enter External Marks (5 courses) for " + name + ": ");
            int[] externalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                System.out.print("Enter Marks: ");
                externalMarks[j] = sc.nextInt();
            }
            sc.nextLine();
            internalStudents[i] = new Internal(name, internalMarks);
            externalStudents[i] = new External(name, externalMarks);
        }
        System.out.println("\nFinal Marks for all students:");
        for (int i = 0; i < n; i++)
        {
            int[] internalMarks = internalStudents[i].getMarks();
            int[] externalMarks = externalStudents[i].getMarks();
            System.out.println("\nStudent: " + internalStudents[i].getName());
            System.out.print("Internal Marks: ");
            for (int mark : internalMarks) {
```

```
            System.out.print(mark + " ");
        }
        System.out.print("\nExternal Marks: ");
        for (int mark : externalMarks) {
            System.out.print(mark + " ");
        }
        System.out.print("\nFinal Marks: ");
        for (int j = 0; j < 5; j++) {
            int finalMark = internalMarks[j] + externalMarks[j];
            System.out.print(finalMark + " ");
        }
        System.out.println();
    }
    sc.close();
  }
}
```

Output:

```
C:\Abhinav 3A\Package>java Main
Enter the number of students: 1
Enter the name of student 1: Abhinav
Enter internal marks (5 courses) for Abhinav:
78
89
98
87
76
Enter external marks (5 courses) for Abhinav:
78
89
87
89
87

Final Marks for all students:

Student: Abhinav
Internal Marks: 78 89 98 87 76
External Marks: 78 89 87 89 87
Final Marks: 156 178 185 176 163
```

Program 7

Exceptions

Algorithm:

1) **Start** the program.
2) **Create a Scanner object** to read user input.
3) **Prompt** the user to enter the following:

- **Son's Name**.
- **Son's Age**.
- **Father's Name**.
- **Father's Age**.

4) **Create a Son object** with the inputted data (son's name, son's age, father's name, and father's age):

- Call the **Father constructor** with the father's name and father's age.
- If the father's age is negative, throw a NegativeAgeError and display the error message.
- Check the son's age:
    - If the son's age is negative, throw a NegativeAgeError and display the error message.
    - If the son's age is greater than or equal to the father's age, throw an InvalidAgeError and display the error message.

5) If no errors occur during the input validation, assign the values of son's name and son's age to the Son object.
6) **Display** the message: "End of program".
7) **Close** the Scanner object to release the resources.
8) **End** the program.

20/11/24

LAB - 7

② Write a program that demonstrates handling of exception in inheritance tree. Create a base class called as "Father" and a derived class called as "Son" which extends the base class. In Father's class implement a constructor which takes the age and throws exception "Wrong age" when input age is less than zero. In Son's class implement a constructor that uses both father's and son's age and throws exception if son's age is greater than or equal to father's age.

```
class NegativeAgeError extends Exception {
int a;
public NegativeAgeError (int a) {
   this.a = a;
}
public String toString () {
   return "Negative Age: " + a ; }
}

class InvalidAgeError extends Exception {
 int a, b ;
 public InvalidAgeError ( int a, int b) {
    this.a = a;
    this.b = b;
}
 public String toString () {
    return "Invalid age: " + a + " is less than " + b ; }
}
```

```
class Father {
  String name;
  int age;
  Father(String name, int age) {
   try {
      if (age < 0) {
         throw new NegativeAgeError(age);
        }
       this.name = name;
       this.age = age;
}
 catch (NegativeAgeError e) {
  this.age = age
 System.out.println(e);}
 }
}
class Son extends Father {
String sonname;
int sonage;
Son(String sonname, int sonage, String fathername,
   int fatherage) {
super(fathername, fatherage);
this.sonname = sonname;
 try {
   if (sonAge < 0) {
      throw new NegativeAgeError(sonAge);
    }
   if (sonAge >= fatherage) {
      throw new InvalidAgeError(sonAge, FatherAge);
   }
```

```
        this.sonAge = sonAge;
    }
    catch (NegativeAgeError e) {
        System.out.println(e);
    }
    catch (InvalidAgeError e) {
        System.out.println(e);}
    }
}
class Exceptions {
    public static void main (String[] args) {
    Scanner sc = new Scanner (system.in);
    System.out.println("Enter son name");
    String name = sc.next();
    System.out.println("Enter son age");
    int age = sc.nextInt();
    System.out.println("Enter father name");
    String name = sc.next();
    System.out.println("Enter father age");
    int age = sc.nextInt();

        Son a1 = new Son(name, age, name, age);
System.out.println("End of program");
    }
}
```

Output :

Enter son name

a

Enter son age

5

Enter father name

b

Enter father age

3

Invalid Age : 5 is less than 3

Seen

Enter son name

a

Enter son age

5

Enter father name

b

Enter father age

-3

Negative Age : - 4

Invalid Age : 5 is less than - 4

Code:

```
import java.util.Scanner;

class NegativeAgeError extends Exception {
    int a;

    public NegativeAgeError(int a) {
        this.a = a;
    }

    public String toString() {
        return "Negative Age: " + a;
    }
}

class InvalidAgeError extends Exception {
    int a, b;

    public InvalidAgeError(int a, int b) {
        this.a = a;
        this.b = b;
    }

    public String toString() {
        return "Invalid Age: " + a + " is less than " + b;
    }
}

class Father {
    String name;
    int age;

    Father(String name, int age) {
        try {
            if (age < 0) {
                throw new NegativeAgeError(age);
            }
            this.name = name;
            this.age = age;
        } catch (NegativeAgeError e) {
            System.out.println(e);
```

```
        }
    }
}

class Son extends Father {
    String sonName;
    int sonAge;

    Son(String sonName, int sonAge, String fatherName, int fatherAge) {
        super(fatherName, fatherAge);
        this.sonName = sonName;
        try {
            if (sonAge < 0) {
                throw new NegativeAgeError(sonAge);
            }
            if (sonAge >= fatherAge) {
                throw new InvalidAgeError(sonAge, fatherAge);
            }
            this.sonAge = sonAge;
        } catch (NegativeAgeError e) {
            System.out.println(e);
        } catch (InvalidAgeError e) {
            System.out.println(e);
        }
    }
}

class Exceptions {
    public static void main(String[] args) {
        System.out.println("Abhinav Sanjay\nUSN : 1BM23CS009");
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter son name");
        String name = sc.next();

        System.out.println("Enter son age");
        int age = sc.nextInt();

        System.out.println("Enter father name");
        String fatherName = sc.next();

        System.out.println("Enter father's age");
```

```
        int fatherAge = sc.nextInt();

        Son a1 = new Son(name, age, fatherName, fatherAge);

        System.out.println("End of program");
        sc.close();
    }
}
```

Output:

```
C:\Abhinav 3A>java Exceptions
Abhinav Sanjay
USN : 1BM23CS009
Enter son name
a
Enter son age
5
Enter father name
b
Enter fathers age
3
Invalid Age: 5 is less than 3

C:\Abhinav 3A>java Exceptions
Abhinav Sanjay
USN : 1BM23CS009
Enter son name
a
Enter son age
5
Enter father name
b
Enter fathers age

-4
Negative Age: -4
Invalid Age: 5 is less than -4
```

Program 8

Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

Algorithm:

1) **Start** the program.

2) **Create a Thread for BMS**:

   - Define a class BMS that extends Thread.
   - In the run() method of BMS class:
     - Continuously print "BMS College of Engineering".
     - Pause the execution for 10 seconds using Thread.sleep(10000).

3) **Create a Thread for CSE**:

   - Define a class CSE that extends Thread.
   - In the run() method of CSE class:
     - Continuously print "CSE".
     - Pause the execution for 2 seconds using Thread.sleep(2000).

4) **Main Method Execution**:

   - Create an object bmsThread of the BMS class and start the thread.
   - Create an object cseThread of the CSE class and start the thread.

5) **Repeat**:

   - Both threads (BMS and CSE) will run concurrently, printing "BMS College of Engineering" every 10 seconds and "CSE" every 2 seconds, respectively.

6) **End** of program execution

27/11/24

LAB - 8

(8) Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another one displaying "CSE" once every 2 seconds.

```
class BMS extends Thread {
  public void run () {
    while ( true ) {
      System. out. println ("BMS College of Engineering");
      try {
        Thread. sleep (10000);
        }
        catch ( InterruptedException e ) {
          System. out. println ( e ); }
        }
    }
}
class CSE extends Thread {
  public void run () {
    while ( true ) {
      System. out. println ( "CSE");
      try {
        Thread. sleep (2000)
        }
        catch ( InterruptedException e ) {
          System. out. println ( e ); }
        }
    }
}
```

```
public class CS_IS_ThreadProgram {
    public static void main (String [] args) {
        BMS bmsThread = new BMS();
        bmsThread.start();

        CSE cseThread = new CSE();
        cseThread.start(); }
}
```

Output:

```
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
```

Seen

27/11/24

Code:

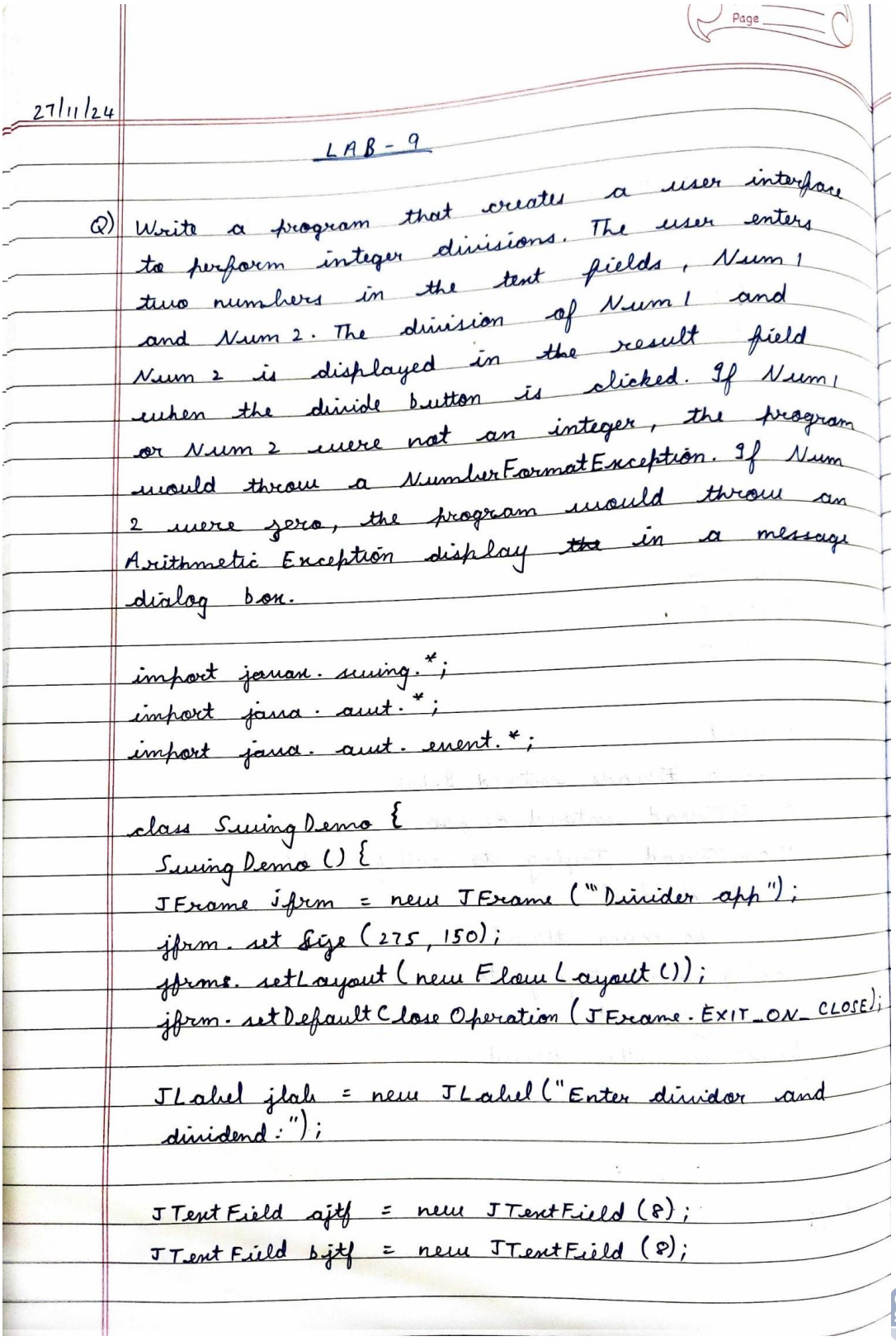
```
class BMS extends Thread {
    public void run() {
        while (true) {
            System.out.println("BMS College of Engineering");
            try {
                Thread.sleep(10000); // Sleep for 10 seconds
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}
class CSE extends Thread {
    public void run() {
        while (true) {
            System.out.println("CSE");
            try {
                Thread.sleep(2000); // Sleep for 2 seconds
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

public class BMSCE {
    public static void main(String[] args) {
        System.out.println("Abhinav Sanjay\nUSN : 1BM23CS009");

        // Create and start the threads
        BMS bmsThread = new BMS();
        bmsThread.start();
        CSE cseThread = new CSE();
        cseThread.start();
    }
}
```

Output:

```
D:\Abhinav3A\Java>java BMSCE
Abhinav Sanjay
USN : 1BM23CS009
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
```

Program 9

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an Arithmetic Exception Display the exception in a message dialog box.



27/11/24

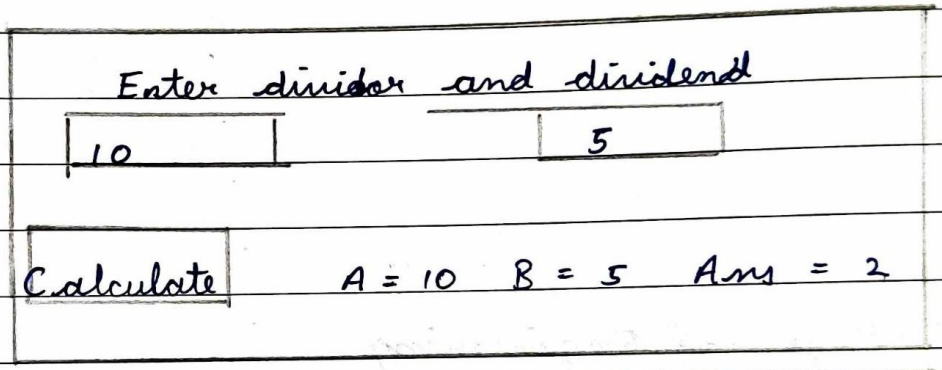
LAB - 9

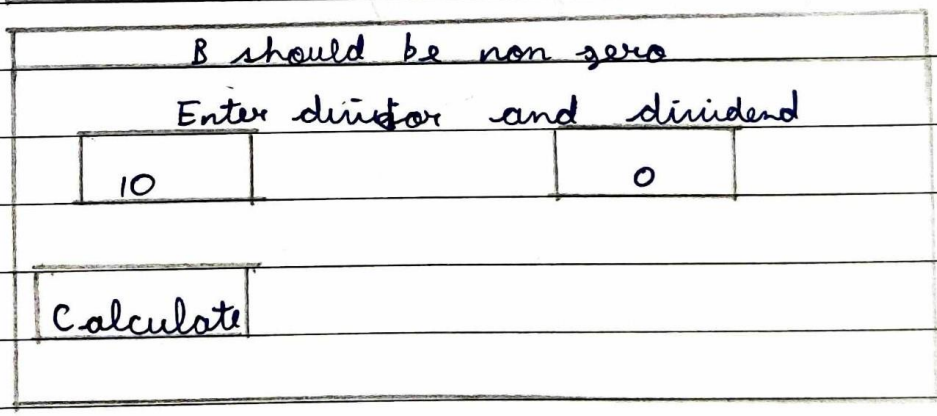
Q) Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the result field when the divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num 2 were zero, the program would throw an Arithmetic Exception display ~~the~~ in a message dialog box.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {
  SwingDemo() {
  JFrame jfrm = new JFrame("Divider app");
  jfrm.setSize(275, 150);
  jfrm.setLayout(new FlowLayout());
  jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  JLabel jlab = new JLabel("Enter divisor and
  dividend:");

  JTextField ajtf = new JTextField(8);
  JTextField bjtf = new JTextField(8);
```

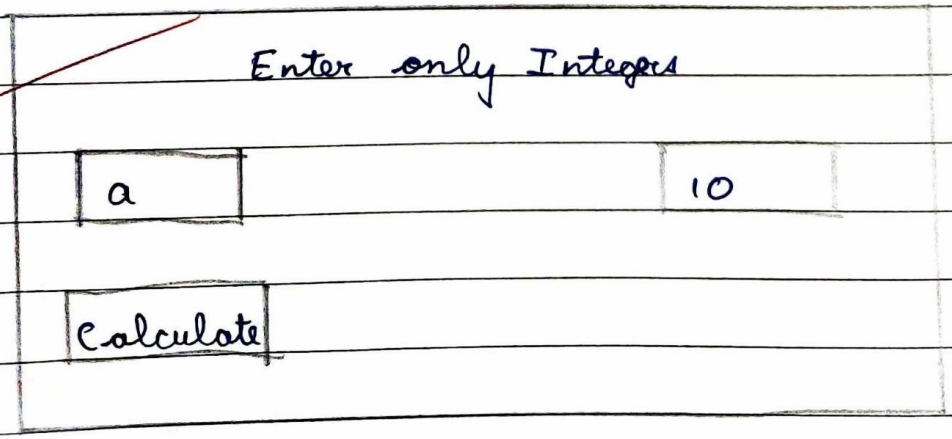
```
JButton button = new JButton ("Calculate");
JLabel err = new JLabel ();
JLabel alab = new JLabel ();
JLabel blab = new JLabel ();
JLabel anslab = new JLabel ();

jfrm.add (err);
jfrm.add (jlab);
jfrm.add (ajtf);
jfrm.add (bjtf);
jfrm.add (button);
jfrm.add (alab);
jfrm.add (blab);
jfrm.add (anslab);

ActionListener I = new ActionListener ();
 public void actionPerformed (ActionEvent evt) {
  System.out.println ("Action event from text field");
  }
};
 ajtf.add ActionListener ();
bjtf.add ActionListener ();
button.add ActionListener (new ActionListener() {
  public Void actionPerformed (ActionEvent evt) {
  try {
     int a = Integer.parseInt (ajtf.getText ());
     int b = Integer.parseInt (bjtf.getText ());
     int ans = a/b;
```

```
alab.setText(" ");
blab.setText(" ");
anslab.setText(" ");
err.setText("Enter only Integers!");
}
catch(ArithmeticException e){
 alab.setText(" ");
 blab.setText(" ");
 anslab.setText(" ");
 err.setText("B should be non zero!");}
 }
});

jfrm.setVisible(true);
}

public static void main(String args[]){
 SwingUtilities.invokeLater(new Runnable(){
  public void run(){
   new SwingDemo();}
 });
 }
}
```





Seen



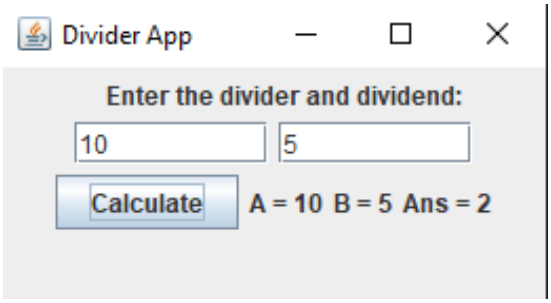
Code:

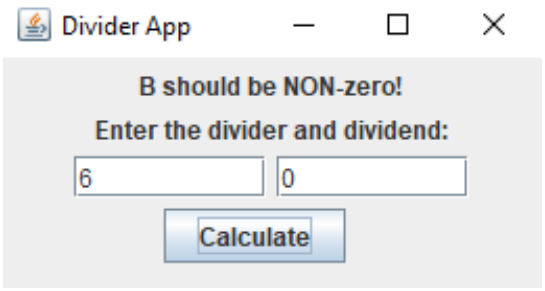
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {

    SwingDemo() {
        // Create JFrame container
        JFrame jfrm = new JFrame("Divider App");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new FlowLayout());

        // To terminate on close
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Text label
        JLabel jlab = new JLabel("Enter the divider and dividend:");

        // Add text fields for both numbers
        JTextField ajtf = new JTextField(8);
        JTextField bjtf = new JTextField(8);

        // Calculate button
        JButton button = new JButton("Calculate");

        // Labels for displaying results
        JLabel err = new JLabel();
        JLabel alab = new JLabel();
        JLabel blab = new JLabel();
        JLabel anslab = new JLabel();

        // Add components to the frame in order
        jfrm.add(err);      // To display error messages
        jfrm.add(jlab);
        jfrm.add(ajtf);
        jfrm.add(bjtf);
        jfrm.add(button);
        jfrm.add(alab);
```

```
jfrm.add(blab);
jfrm.add(anslab);

// Action listener for the text fields (ajtf and bjtf)
ActionListener l = new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      System.out.println("Action event from a text field");
   }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);

// Action listener for the calculate button
button.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      try {
         // Get values from text fields and perform division
         int a = Integer.parseInt(ajtf.getText());
         int b = Integer.parseInt(bjtf.getText());
         int ans = a / b;

         alab.setText("\nA = " + a);
         blab.setText("\nB = " + b);
         anslab.setText("\nAns = " + ans);
      } catch (NumberFormatException e) {
         // Handle non-integer input
         alab.setText("");
         blab.setText("");
         anslab.setText("");
         err.setText("Enter Only Integers!");
      } catch (ArithmeticException e) {
         // Handle division by zero
         alab.setText("");
         blab.setText("");
         anslab.setText("");
         err.setText("B should be NON-zero!");
      }
   }
});
```

```
        // Display frame
        jfrm.setVisible(true);
    }

    public static void main(String args[]) {
        // Create frame on event dispatching thread
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new SwingDemo();
            }
        });
    }
}
```

Output:





Program 10

Demonstrate Inter process Communication and deadlock

27/11/24

LAB - 10

Q) Demonstrate inter process Communication and deadlock.

```
class Q {
 int n;
 boolean valueSet = false;
 synchronized int get() {
 while (!valueSet)
  try {
   System.out.println("In Consumer Waiting\n");
   wait();
  }
  catch (InterruptedException e) {
   System.out.println("InterruptedException caught");
  }
System.out.println("Got: " + n);
valueSet = false;
System.out.println("In Intimate Producer\n");
notify();
return n;
}
synchronized void put(int n) {
  while (valueSet)
  try {
   System.out.println("Producer Waiting");
   wait();
  }
  catch (InterruptedException e) {
   System.out.println("InterruptedException caught");
  }
```

```
this.n = n;
valueSet = true;
System.out.println("Put : " + n);
System.out.println("Intimate Customer");
notify(); }
}

class Producer implements Runnable {
Q q;
Producer(Q, q) {
    this.q = q;
    new Threads(this, "Producer").start();
}
public void run() {
    int i = 0;
    while (i < 15) {
    q.put(i++); }
    }
}

class Consumer implements Runnable {
  Q q;
  Consumer(Q, q) {
     this.q = q;
     new Thread(this, "Consumer").start();
  }
```

```
public void run () {
    int i = 0 ;
    while (i <15) {
        int r = q. get ();
        System. out. println (" consumed :" + r);
        i ++ ; }
    }
}

class PCFixed {
    public static void main (String args [ ]) {
    Q q = new Q ();
    new Producer (q);
    new Consumer (q);
    System. out. println ("Press Control-C to stop");
    }
}
```

Put : 1

Get : 1

Put : 2

Get : 2

Put : 3

Get : 3

Put : 4

Get : 4

Put : 5

Get : 5

27/11/24

Deadlock

```
class A {
  synchronized void foo (B b) {
  String name = Thread.currentThread().getName();
  System.out.println(name + " entered A.foo");
  try {
   Thread.sleep(1000); }
   catch (Exception e) {
    System.out.println("A Interrupted");
   }
  System.out.println(name + " trying to call
  B.last()");
  b.last();
  }
  void last() {
   System.out.println("Inside A.last"); }
}

class B {
  synchronized void bar (A a) {
   String name = Thread.currentThread().getName();
   System.out.println(name + " entered B.bar");
   try {
      Thread.sleep(1000); }
   catch ( Exception e ) {
   System.out.println("B Interrupted");
   }
```

```
System.out.println(name + " trying to call A.last()");
a.last();
}

void last() {
 System.out.println("Inside A.last");}
}
class Deadlock implements Runnable {
 A a = new A();
 B b = new B();
 Deadlock() {
  Thread.currentThread().setName("MainThread");
  Thread t = new Thread(this, "RacingThread");
  t.start();
  a.foo(b);
  System.out.println("Back in main thread");}

public void run() {
 b.bar(a);
 System.out.println("Back in other thread");
 }
 public static void main(String args[]) {
   new Deadlock(); }
}
```

Output :

Racing Threads entered B.bar

Main Thread entered A.foo

Main Thread trying to call B.last()

Inside A.last

Back in main thread

Racing Thread trying to call A.last()

Inside A.last

Back in other thread

Seen

27/4/24

Code for Inter Process Communication

```
class Q {

   int n;
   boolean valueSet = false;

   synchronized int get() {
      while (!valueSet) {
         try {
            System.out.println("\nConsumer waiting\n");
            wait();
         } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
         }
      }

      System.out.println("Got: " + n);
      valueSet = false;

      System.out.println("\nIntimate Producer\n");
      notify();

      return n;
   }

   synchronized void put(int n) {
      while (valueSet) {
         try {
            System.out.println("\nProducer waiting\n");
            wait();
         } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
         }
      }
```

```
        this.n = n;
        valueSet = true;

        System.out.println("Put: " + n);
        System.out.println("\nIntimate Consumer\n");
        notify();
    }
}

class Producer implements Runnable {

    Q q;

    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run() {
        int i = 0;
        while (i < 15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {

    Q q;

    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
```

```
    public void run() {
        int i = 0;
        while (i < 15) {
            int r = q.get();
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}

class PCFixed {

    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

Output

```
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer


Producer waiting

consumed:0
Got: 1

Intimate Producer

consumed:1
Put: 2

Intimate Consumer


Producer waiting

Got: 2

Intimate Producer

consumed:2
Put: 3

Intimate Consumer
```

```
Producer waiting

Got: 3

Intimate Producer

consumed:3
Put: 4

Intimate Consumer


Producer waiting

Got: 4

Intimate Producer

consumed:4
Put: 5

Intimate Consumer


Producer waiting

Got: 5

Intimate Producer

consumed:5
```

Code for Deadlock

```
class A {

   synchronized void foo(B b) {
      String name = Thread.currentThread().getName();
      System.out.println(name + " entered A.foo");

      try {
         Thread.sleep(1000);
      } catch (Exception e) {
         System.out.println("A Interrupted");
      }

      System.out.println(name + " trying to call B.last()");
      b.last();
   }

   void last() {
      System.out.println("Inside A.last");
   }
}

class B {

   synchronized void bar(A a) {
      String name = Thread.currentThread().getName();
      System.out.println(name + " entered B.bar");

      try {
         Thread.sleep(1000);
      } catch (Exception e) {
         System.out.println("B Interrupted");
      }

      System.out.println(name + " trying to call A.last()");
      a.last();
   }
```

```
    void last() {
        System.out.println("Inside B.last");
    }
}

class Deadlock implements Runnable {

    A a = new A();
    B b = new B();

    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();

        a.foo(b); // get lock on a in this thread.
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a); // get lock on b in other thread.
        System.out.println("Back in other thread");
    }

    public static void main(String args[]) {
        new Deadlock();
    }
}
```

Output:

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
Inside A.last
Back in other thread
```